Problèmes de décision et d'évaluation en complexité algébrique

Sylvain Perifel

Thèse encadrée par Pascal Koiran

LIP, ENS Lyon

Le 6 décembre 2007

Decision and Evaluation Problems in Algebraic Complexity

Sylvain Perifel

Thesis supervised by Pascal Koiran

LIP, ENS Lyon

December 6, 2007

Title explanation

- Complexity: classification of problems with respect to the minimal number of steps required to solve them.
  Examples: 1. Cooking pasta $\leq$ Making a mille-feuille.

Title explanation

- Complexity: classification of problems with respect to the minimal number of steps required to solve them.

  Examples:
  1. Cooking pasta $\leq$ Making a mille-feuille.
  2. Determinant $\leq$ Permanent.

Title explanation

- Complexity: classification of problems with respect to the minimal number of steps required to solve them.
  Examples: 1. Cooking pasta $\leq$ Making a mille-feuille.
  2. Determinant $\leq$ Permanent.

- Algebraic: polynomials, rich mathematical structures (fields).
  Example: Minimal number of operations to compute $P(x) = \prod_{i=0}^{n}(x - i)$?

Title explanation

- Complexity: classification of problems with respect to the minimal number of steps required to solve them.
  Examples: 1. Cooking pasta $\leq$ Making a mille-feuille.
  2. Determinant $\leq$ Permanent.

- Algebraic: polynomials, rich mathematical structures (fields).
  Example: Minimal number of operations to compute $P(x) = \prod_{i=0}^{n}(x - i)$?

- Decision problem: yes/no answer.
  Examples: 1. Did I eat 5 mille-feuilles?

Title explanation

- Complexity: classification of problems with respect to the minimal number of steps required to solve them.
  Examples: 1. Cooking pasta $\leq$ Making a mille-feuille.
  2. Determinant $\leq$ Permanent.

- Algebraic: polynomials, rich mathematical structures (fields).
  Example: Minimal number of operations to compute $P(x) = \prod_{i=0}^{n}(x - i)$?

- Decision problem: yes/no answer.
  Examples: 1. Did I eat 5 mille-feuilles?
  2. Given x, is it a root of P?

Title explanation

- Complexity: classification of problems with respect to the minimal number of steps required to solve them.
  Examples: 1. Cooking pasta $\leq$ Making a mille-feuille.
  2. Determinant $\leq$ Permanent.

- Algebraic: polynomials, rich mathematical structures (fields).
  Example: Minimal number of operations to compute $P(x) = \prod_{i=0}^{n}(x - i)$?

- Decision problem: yes/no answer.
  Examples: 1. Did I eat 5 mille-feuilles?
  2. Given x, is it a root of P?

- Evaluation problem: computing a result.
  Examples: 1. How many mille-feuilles did I eat?

Title explanation

- Complexity: classification of problems with respect to the minimal number of steps required to solve them.
  Examples: 1. Cooking pasta $\leq$ Making a mille-feuille.
  2. Determinant $\leq$ Permanent.

- Algebraic: polynomials, rich mathematical structures (fields).
  Example: Minimal number of operations to compute $P(x) = \prod_{i=0}^{n}(x - i)$?

- Decision problem: yes/no answer.
  Examples: 1. Did I eat 5 mille-feuilles?
  2. Given x, is it a root of P?

- Evaluation problem: computing a result.
  Examples: 1. How many mille-feuilles did I eat?
  2. Compute the polynomial P.

Outline

Outline

Boolean complexity

- Boolean (discrete) complexity: problems on words over the alphabet $\{0, 1\}$.

Boolean complexity

- Boolean (discrete) complexity: problems on words over the alphabet $\{0, 1\}$.
  $\rightarrow$ Coding the objects in binary (integers, graphs, ...).
  Example: Given a graph G, is there a path visiting every vertex exactly once?

Boolean complexity

- Boolean (discrete) complexity: problems on words over the alphabet $\{0, 1\}$.
  $\rightarrow$ Coding the objects in binary (integers, graphs, ...).
  Example: Given a graph G, is there a path visiting every vertex exactly once?

- Turing machines (or Boolean circuits), polynomial time P, nondeterministic polynomial time NP (verifying instead of finding).
  Question $\mathrm{P} = \mathrm{NP}$? $\rightarrow$ NP-completeness.

Boolean complexity

- Boolean (discrete) complexity: problems on words over the alphabet $\{0, 1\}$.
  $\rightarrow$ Coding the objects in binary (integers, graphs, ...).
  Example: Given a graph G, is there a path visiting every vertex exactly once?

- Turing machines (or Boolean circuits), polynomial time P, nondeterministic polynomial time NP (verifying instead of finding).
  Question $\mathrm{P} = \mathrm{NP}$? $\rightarrow$ NP-completeness.

  ... But:
  1. Main open questions remain elusive.

Boolean complexity

- Boolean (discrete) complexity: problems on words over the alphabet $\{0, 1\}$.
  $\rightarrow$ Coding the objects in binary (integers, graphs, ...).
  Example: Given a graph G, is there a path visiting every vertex exactly once?

- Turing machines (or Boolean circuits), polynomial time P, nondeterministic polynomial time NP (verifying instead of finding).
  Question $\mathrm{P} = \mathrm{NP}$? $\rightarrow$ NP-completeness.

  ... But:
  1. Main open questions remain elusive.
  2. Discrete setting not adapted to algebraic algorithms.

Algebraic complexity

- Algebraic complexity: problems over e.g. $\mathbb{R}$ or $\mathbb{C}$.
  - More adapted to algebraic problems.

Algebraic complexity

- Algebraic complexity: problems over e.g. $\mathbb{R}$ or $\mathbb{C}$.
  - More adapted to algebraic problems.
  - Richer structure $\rightarrow$ more tools for fundamental questions?

Algebraic complexity

- Algebraic complexity: problems over e.g. $\mathbb{R}$ or $\mathbb{C}$.
  - More adapted to algebraic problems.
  - Richer structure $\rightarrow$ more tools for fundamental questions?

Examples: 1. Given a system of polynomials, does it have a solution?

Algebraic complexity

- Algebraic complexity: problems over e.g. $\mathbb{R}$ or $\mathbb{C}$.
  - More adapted to algebraic problems.
  - Richer structure $\rightarrow$ more tools for fundamental questions?

Examples:

1. Given a system of polynomials, does it have a solution?
2. Compute the determinant of a matrix.

Algebraic complexity

- Algebraic complexity: problems over e.g. $\mathbb{R}$ or $\mathbb{C}$.
  - More adapted to algebraic problems.
  - Richer structure $\rightarrow$ more tools for fundamental questions?

  Examples:
  1. Given a system of polynomials, does it have a solution?
  2. Compute the determinant of a matrix.

- Algebraic circuits (decision problems) / arithmetic circuits (evaluation problems).
- Classes P and NP, NP-completeness...

Overview of Valiant's and BSS models

- Valiant's model: families of polynomials.
  Number of operations $+$ and $\times$ to compute them.

Overview of Valiant's and BSS models

- Valiant's model: families of polynomials.
  Number of operations $+$ and $\times$ to compute them.

- Blum, Shub, and Smale (BSS) model: decision problems (languages) over $\mathbb{R}$ or $\mathbb{C}$.
  Number of operations $+$, $\times$ and $\leq$ (or $=$) to decide them.

Overview of Valiant's and BSS models

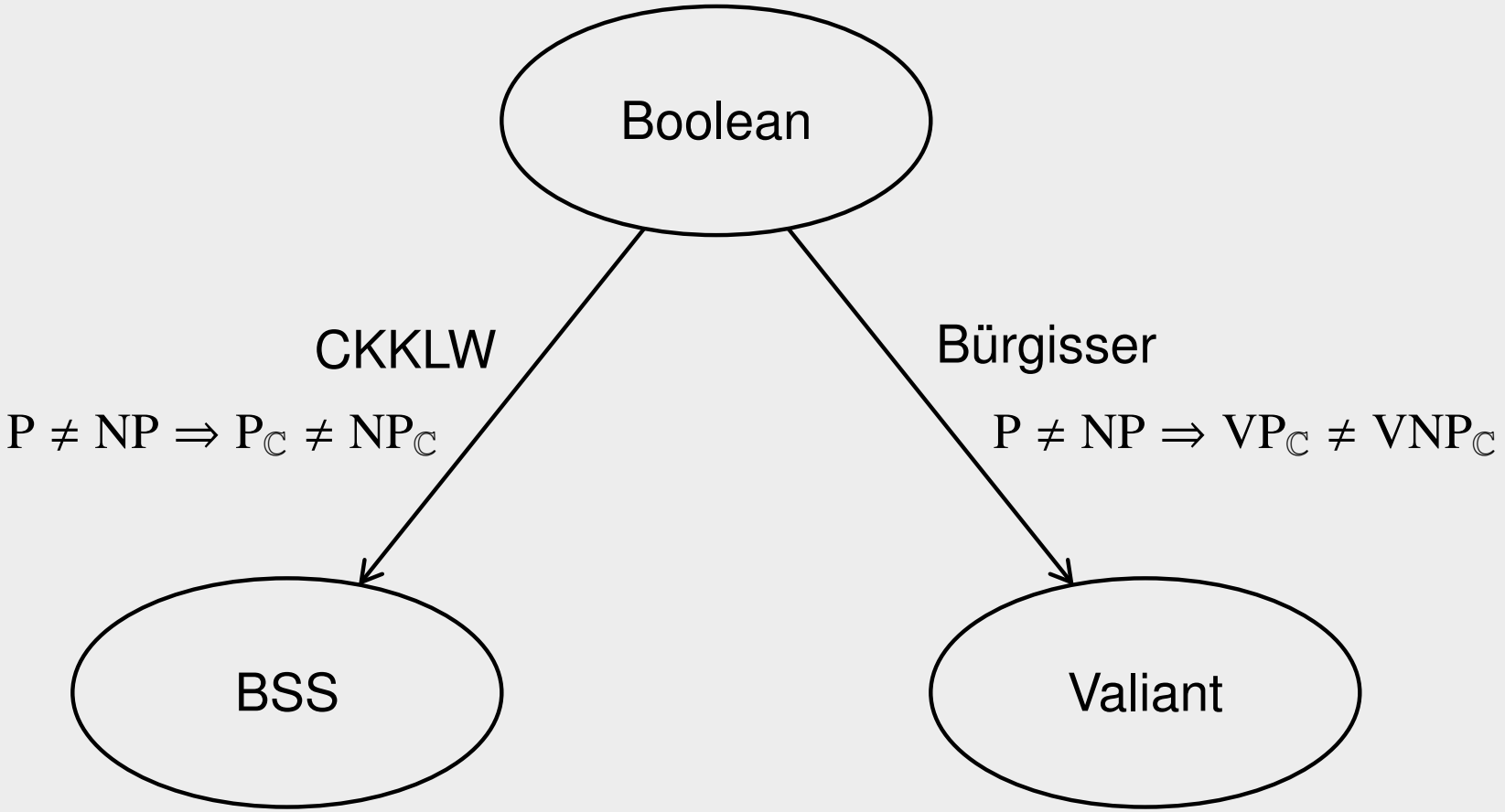
- Valiant's model: families of polynomials.
  Number of operations $+$ and $\times$ to compute them.

- Blum, Shub, and Smale (BSS) model: decision problems (languages) over $\mathbb{R}$ or $\mathbb{C}$.
  Number of operations $+$, $\times$ and $\leq$ (or $=$) to decide them.

- Remark: In this talk we won't bother with the issue of constants and uniformity.

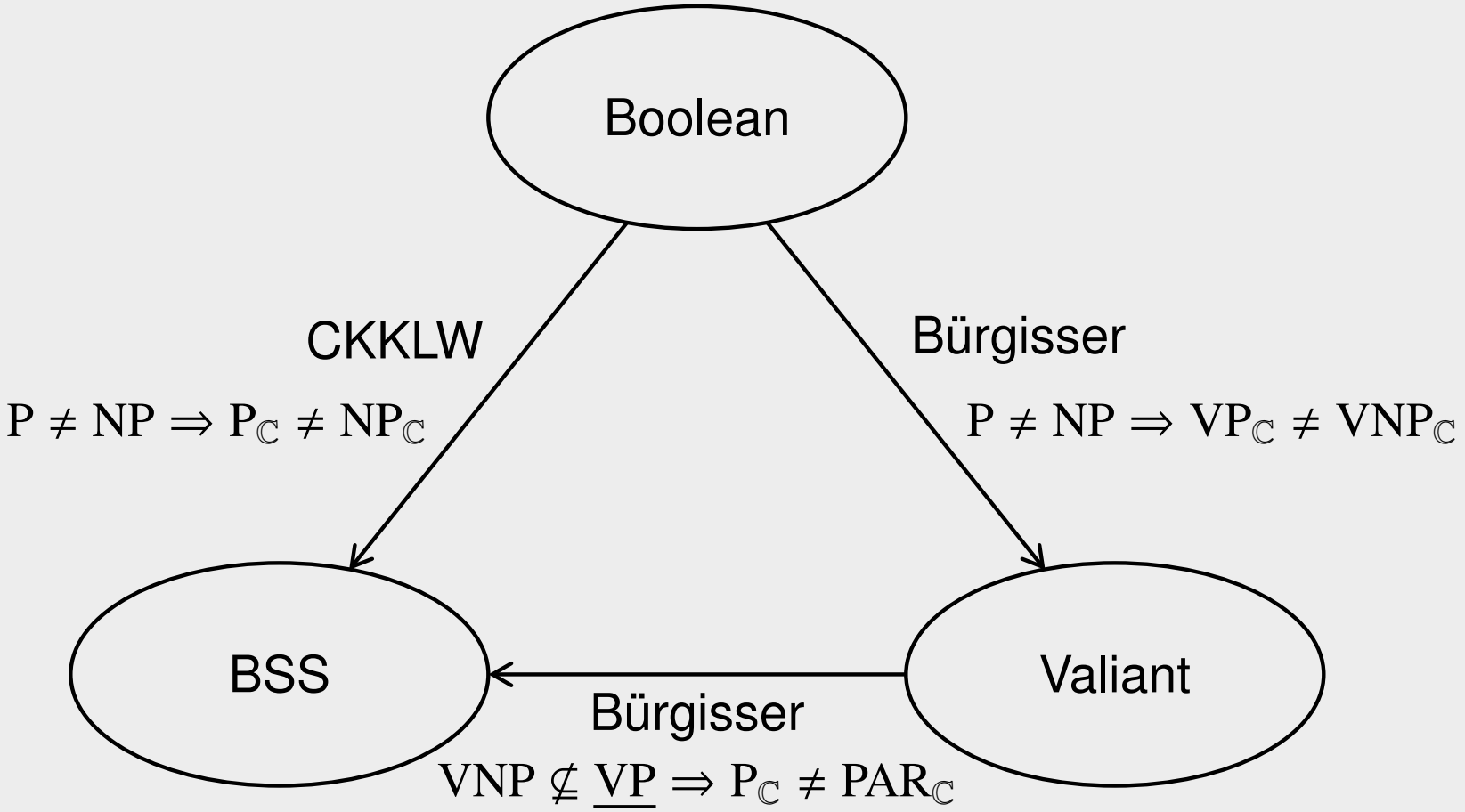
Known implications

The question $\mathrm{P} \neq \mathrm{NP}$ in different models.



Known implications

The question $\mathrm{P} \neq \mathrm{NP}$ in different models.

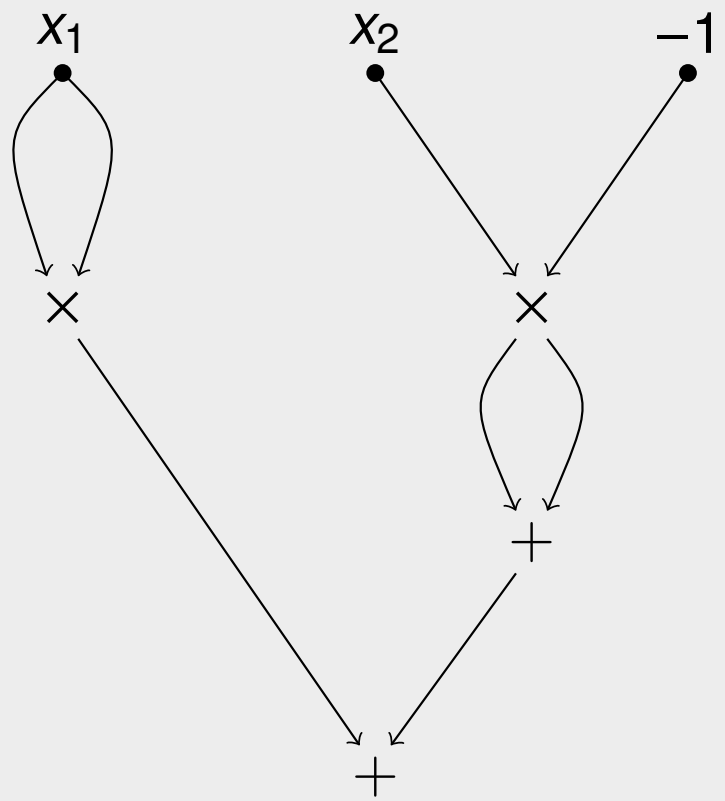


Outline

Arithmetic circuits

Arithmetic circuits:

- gates $+$ and $\times$;
- inputs $x_1, \ldots, x_n$ and constants $\alpha \in K$;
- $\rightarrow$ multivariate polynomials with coefficients in K.



Examples

Polynomials of
exponential degree

Examples

Polynomials of exponential degree

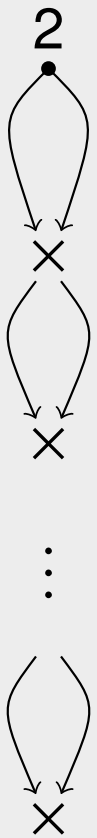
Integers of exponential size

Examples

Polynomials of exponential degree

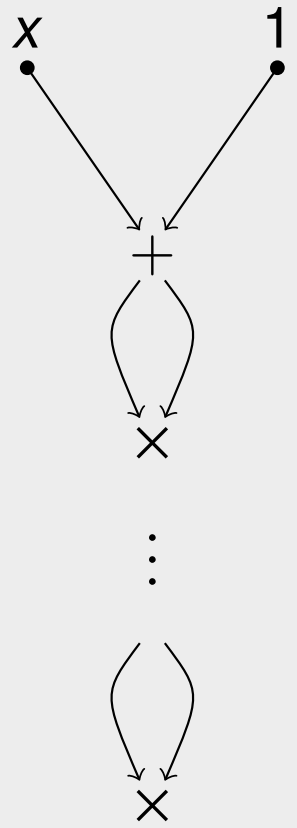
x^{2^n}

Integers of exponential size



2^{2^n}

Hard to compute coefficients



coeff. $\binom{2^n}{k}$

Polynomials given by circuits (1)

Polynomials given by arithmetic circuits are not easily handled by (Boolean) Turing machines.

Polynomials given by circuits (1)

Polynomials given by arithmetic circuits are not easily handled by (Boolean) Turing machines.

- CMN: Given a circuit C and a monomial m, does m appear in the polynomial computed by C?
- DEG: Given a circuit C, compute the degree of the polynomial computed by C.

Polynomials given by circuits (2)

Boolean complexity of the problems DEG and CMN.

The proofs rely on results of Malod (2003) and Allender, Bürgisser, Kjeldgaard-Pedersen, Miltersen (2006).

	Characteristic zero		Characteristic $k > 0$	
	Lower bound	Upper bound	Low. bound	Up. bound
CMN	$\mathrm{P}^{\sharp\mathrm{P}}$	CH	$\mathrm{Mod}_k\mathrm{P}$	
DEG	P	CH	P	$\mathrm{coNP}^{\mathrm{Mod}_k\mathrm{P}}$

P in Valiant's model

P in Valiant's model

- Family of polynomials (f_n): one arithmetic circuit C_n per polynomial $f_n \in K[x_1, \ldots, x_{u(n)}]$.

P in Valiant's model

- Family of polynomials (f_n): one arithmetic circuit C_n per polynomial $f_n \in K[x_1, \ldots, x_{u(n)}]$.

- VP: families of polynomials of polynomial degree computed by arithmetic circuits of polynomial size.
  Example: the determinant

$$\det_n(x_{1,1}, \ldots, x_{1,n}, x_{2,1}, \ldots, x_{n,n}) = \sum_{\sigma \in \mathcal{S}_n} \epsilon(\sigma) \prod_{i=1}^{n} x_{i,\sigma(i)}.$$

- VNP: exponential sum of a VP family.
  $(g_n) \in \mathrm{VNP}$ if there exists $(f_n(x_1, \ldots, x_{u(n)}, y_1, \ldots, y_{p(n)})) \in \mathrm{VP}$ such that

$$g_n(x_1, \ldots, x_{u(n)}) = \sum_{\bar{\epsilon} \in \{0,1\}^{p(n)}} f_n(\bar{x}, \bar{\epsilon})$$

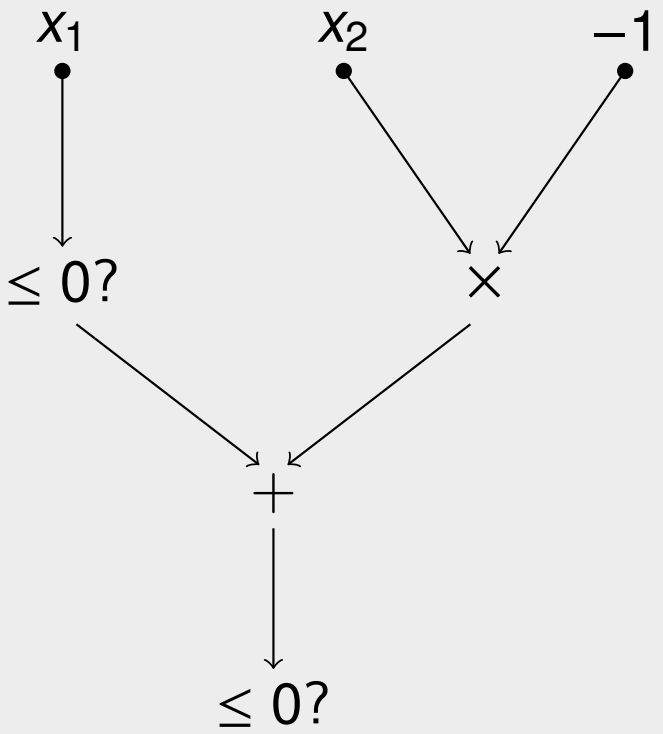
  Example: the permanent (VNP-complete)

$$\mathrm{per}_n(x_{1,1}, \ldots, x_{1,n}, x_{2,1}, \ldots, x_{n,n}) = \sum_{\sigma \in \mathcal{S}_n} \prod_{i=1}^{n} x_{i,\sigma(i)}.$$

Algebraic circuits

Algebraic circuits:

- gates $+$, $\times$ and tests "≤ 0?" (or "$= 0$?", value 0 or 1);
- inputs $x_1, \ldots, x_n$ and constants $\alpha \in K$;
- $\rightarrow$ semialgebraic subset $\{\bar{x} \in K^n \mid C_n(\bar{x}) = 1\}$ of K^n.



P and NP in BSS model

- Families (C_n) of algebraic circuits: C_n has n inputs.
- Language $A \subseteq \cup_{n \geq 0} K^n$ decided by a family (C_n) of algebraic circuits: $x \in A \Leftrightarrow C_{|x|}(x) = 1$.

P and NP in BSS model

- Families (C_n) of algebraic circuits: C_n has n inputs.
- Language $A \subseteq \cup_{n\geq 0} K^n$ decided by a family (C_n) of algebraic circuits: $x \in A \Leftrightarrow C_{|x|}(x) = 1$.

- P_K: sets $A \subseteq \cup_{n\geq 0} K^n$ recognized by circuits of polynomial size.

P and NP in BSS model

- Families (C_n) of algebraic circuits: C_n has n inputs.
- Language $A \subseteq \cup_{n\geq 0} K^n$ decided by a family (C_n) of algebraic circuits: $x \in A \Leftrightarrow C_{|x|}(x) = 1$.

- P_K: sets $A \subseteq \cup_{n\geq 0} K^n$ recognized by circuits of polynomial size.

- NP_K: existential version of P_K, i.e. $A \in \mathrm{NP}_K$ if there exists $B \in \mathrm{P}_K$ such that

$$x \in A \Leftrightarrow \exists y \in K^{p(|x|)} \quad (x, y) \in B.$$

Transfer results

Main open questions in complexity: separations of classes (P and NP, P and $\mathrm{PSPACE}\ldots$).

Transfer results

Main open questions in complexity: separations of classes (P and NP, P and PSPACE...).

- Cucker, Karpinski, Koiran, Lickteig, Werther:
  Boolean separations are harder than in BSS model.

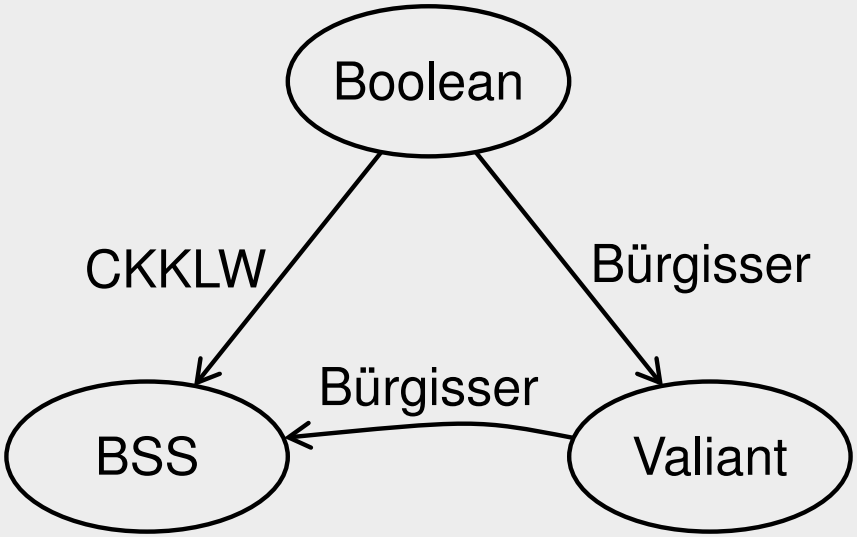
Transfer results

Main open questions in complexity: separations of classes (P and NP, P and $\mathrm{PSPACE}\ldots$).

- Cucker, Karpinski, Koiran, Lickteig, Werther:
  Boolean separations are harder than in BSS model.
- Bürgisser:
  Boolean separations are harder than in Valiant's model.
  Strong separations in Valiant's model imply weak ones in BSS.

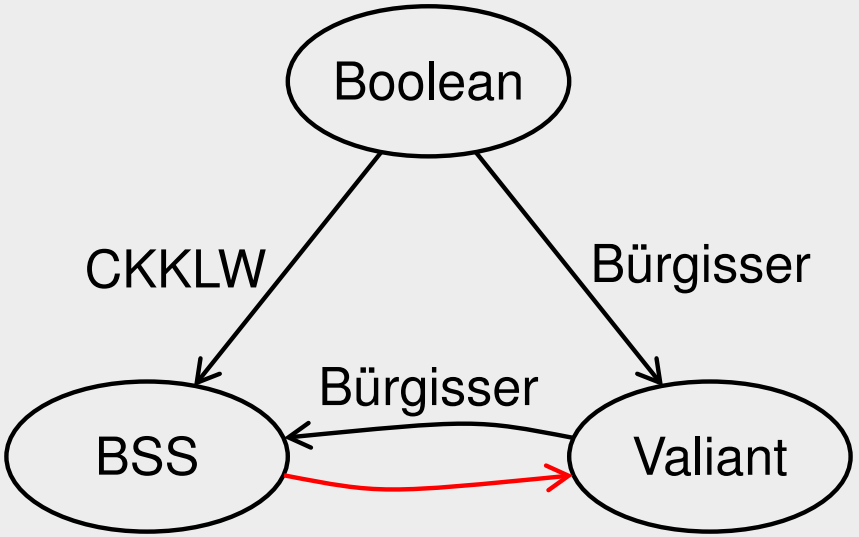
Transfer results

Main open questions in complexity: separations of classes (P and NP, P and PSPACE...).



Transfer results

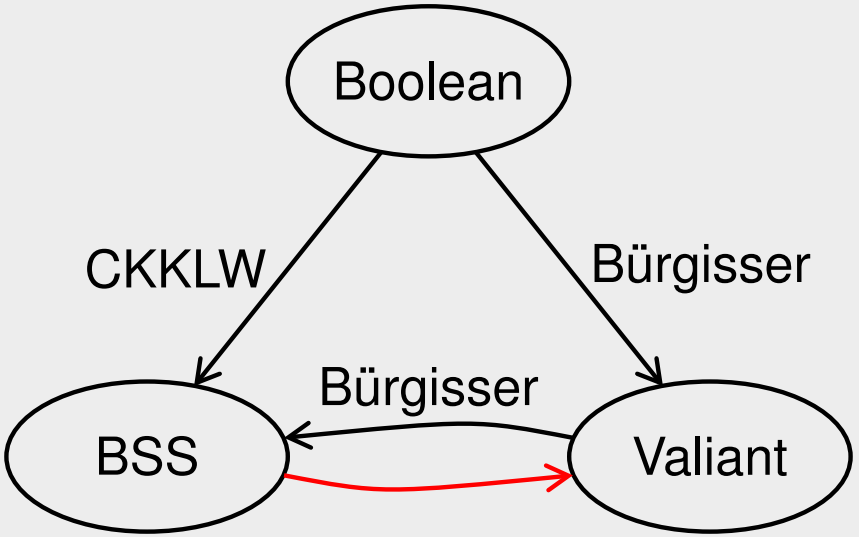
Main open questions in complexity: separations of classes (P and NP, P and PSPACE...).



- Theorem: separations in BSS are harder than in Valiant.

Transfer results

Main open questions in complexity: separations of classes (P and NP, P and $\mathrm{PSPACE}\ldots$).



- Theorem: separations in BSS are harder than in Valiant.

Valiant's model: easier model, easier questions
→ why not begin by its study?

Outline

Twenty Questions

- Problem introduced by Shub and Smale (1995).
  Simple candidate to be in $\mathrm{NP}_{\mathbb{C}} \setminus \mathrm{P}_{\mathbb{C}}$.

Twenty Questions

- Problem introduced by Shub and Smale (1995).
  Simple candidate to be in $\mathrm{NP}_{\mathbb{C}} \setminus \mathrm{P}_{\mathbb{C}}$.
- TwentyQuestions: Given $x \in \mathbb{C}$ and $n \in \mathbb{N}$ (in unary), decide whether $x \in \{0, 1, 2, \ldots, 2^n - 1\}$.

Twenty Questions

- Problem introduced by Shub and Smale (1995).
  Simple candidate to be in $\mathrm{NP}_{\mathbb{C}} \setminus \mathrm{P}_{\mathbb{C}}$.
- TwentyQuestions: Given $x \in \mathbb{C}$ and $n \in \mathbb{N}$ (in unary), decide whether $x \in \{0, 1, 2, \ldots, 2^n - 1\}$.

- If we can compute in polynomial time

$$P(X) = \prod_{i=0}^{2^n-1} (X - i),$$

  then we can decide TwentyQuestions in polynomial time.

Twenty Questions

- Problem introduced by Shub and Smale (1995).
  Simple candidate to be in $\mathrm{NP}_{\mathbb{C}} \setminus \mathrm{P}_{\mathbb{C}}$.
- TwentyQuestions: Given $x \in \mathbb{C}$ and $n \in \mathbb{N}$ (in unary), decide whether $x \in \{0, 1, 2, \ldots, 2^n - 1\}$.

- If we can compute in polynomial time

$$P(X) = \prod_{i=0}^{2^n-1} (X - i),$$

  then we can decide TwentyQuestions in polynomial time.

- $\rightarrow$ If we can compute big products efficiently then we can decide TwentyQuestions efficiently.

NP without multiplication

Can we generalize this result to other problems than TwentyQuestions?

NP without multiplication

Can we generalize this result to other problems than TwentyQuestions?

Yes, to whole $\mathrm{NP}_{(K,+,-,=)}$ (NP without multiplication, contains in particular TwentyQuestions and SubsetSum).

Theorem

If exponential products are computable by polynomial-size circuits, then $\mathrm{NP}_{(K,+,-,=)} \subseteq \mathrm{P}_{(K,+,\times,=)}$.

K is a field of characteristic zero.

Proof (1)

Take a language $A \in \mathrm{NP}_{(K,+,-,=)}$. We want to show that $A \in \mathrm{P}_{(K,+,\times,=)}$ provided we can compute exponential products in polynomial time.

Proof (1)

Take a language $A \in \mathrm{NP}_{(K,+,-,=)}$. We want to show that $A \in \mathrm{P}_{(K,+,\times,=)}$ provided we can compute exponential products in polynomial time.

- By Koiran (1994), existential quantification can be only Boolean:

$$\bar{x} \in A \Leftrightarrow \exists \bar{y} \in \{0,1\}^{p(|\bar{x}|)} \quad (x,y) \in B$$

where $B \in \mathrm{P}_{(K,+,-,=)}$ and p is a polynomial.

Proof (1)

Take a language $A \in \mathrm{NP}_{(K,+,-,=)}$. We want to show that $A \in \mathrm{P}_{(K,+,\times,=)}$ provided we can compute exponential products in polynomial time.

- By Koiran (1994), existential quantification can be only Boolean:

$$\bar{x} \in A \Leftrightarrow \exists \bar{y} \in \{0,1\}^{p(|\bar{x}|)} \quad (x,y) \in B$$

  where $B \in \mathrm{P}_{(K,+,-,=)}$ and p is a polynomial.

- Let $f_1, \ldots, f_s$ be all possible affine functions tested in the circuit for B. The cell of $\bar{x} \in K^n$ consists of the points $\bar{y} \in K^n$ such that $\forall i, f_i(\bar{x}) = 0 \Leftrightarrow f_i(\bar{y}) = 0$.

Proof (1)

Take a language $A \in \mathrm{NP}_{(K,+,-,=)}$. We want to show that $A \in \mathrm{P}_{(K,+,\times,=)}$ provided we can compute exponential products in polynomial time.

- By Koiran (1994), existential quantification can be only Boolean:

$$\bar{x} \in A \Leftrightarrow \exists \bar{y} \in \{0,1\}^{p(|\bar{x}|)} \quad (x,y) \in B$$

  where $B \in \mathrm{P}_{(K,+,-,=)}$ and p is a polynomial.

- Let $f_1, \ldots, f_s$ be all possible affine functions tested in the circuit for B. The cell of $\bar{x} \in K^n$ consists of the points $\bar{y} \in K^n$ such that $\forall i, f_i(\bar{x}) = 0 \Leftrightarrow f_i(\bar{y}) = 0$.

  $\rightarrow$ If $\bar{x}$ and $\bar{y}$ are in the same cell, then they are either both in A or both outside of A.

Proof (2)

Therefore it is enough to find the cell of $\bar{x}$.

- The cell is given by at most n independent hyperplanes to which $\bar{x}$ belongs.
- We use a binary search to find them. Performing this search is done thanks to big products (which replace the existential quantifier in "$\exists i < j \quad \bar{x} \in H_i$?").

Proof (2)

Therefore it is enough to find the cell of $\bar{x}$.

- The cell is given by at most n independent hyperplanes to which $\bar{x}$ belongs.
- We use a binary search to find them. Performing this search is done thanks to big products (which replace the existential quantifier in "$\exists i < j \quad \bar{x} \in H_i$?").
- Once the cell of $\bar{x}$ is found, we can simulate the circuit. □

Proof (2)

Therefore it is enough to find the cell of $\bar{x}$.

- The cell is given by at most n independent hyperplanes to which $\bar{x}$ belongs.
- We use a binary search to find them. Performing this search is done thanks to big products (which replace the existential quantifier in "$\exists i < j \quad \bar{x} \in H_i$?").
- Once the cell of $\bar{x}$ is found, we can simulate the circuit. □

$\rightarrow$ Manipulation of Boolean object (cell) instead of the algebraic input $\bar{x}$.

Proof (2)

Therefore it is enough to find the cell of $\bar{x}$.

- The cell is given by at most n independent hyperplanes to which $\bar{x}$ belongs.
- We use a binary search to find them. Performing this search is done thanks to big products (which replace the existential quantifier in "$\exists i < j \quad \bar{x} \in H_i$?").
- Once the cell of $\bar{x}$ is found, we can simulate the circuit. □

$\rightarrow$ Manipulation of Boolean object (cell) instead of the algebraic input $\bar{x}$.

$\rightarrow$ Elimination of test gates thanks to the simulation of algebraic circuits by Boolean ones, and in turn by arithmetic ones.

Proof (2)

Therefore it is enough to find the cell of $\bar{x}$.

- The cell is given by at most n independent hyperplanes to which $\bar{x}$ belongs.
- We use a binary search to find them. Performing this search is done thanks to big products (which replace the existential quantifier in "$\exists i < j \quad \bar{x} \in H_i$?").
- Once the cell of $\bar{x}$ is found, we can simulate the circuit. □

$\rightarrow$ Manipulation of Boolean object (cell) instead of the algebraic input $\bar{x}$.

$\rightarrow$ Elimination of test gates thanks to the simulation of algebraic circuits by Boolean ones, and in turn by arithmetic ones.

$\rightarrow$ Exponential products replace Boolean existential quantification.

VP and VNP (1)

Theorem

If $\mathrm{VP} = \mathrm{VNP}$ *then exponential products have polynomial-size circuits.*

Proof.

- Lagrange interpolation enables to express a polynomial as an exponential sum (with hard to compute coefficients, though).
- Bürgisser (2006) shows that, in the case of big products, these coefficients are computable in the counting hierarchy.
- If $\mathrm{VP} = \mathrm{VNP}$ then CH collapses and we obtain small circuits for computing big products. □

VP and VNP (2)

Corollary

If $\mathrm{VP} = \mathrm{VNP}$ *then* $\mathrm{NP}_{(K,+,-,=)} \subseteq \mathrm{P}_{(K,+,\times,=)}$.

VP and VNP (2)

Corollary

If $\mathrm{VP} = \mathrm{VNP}$ *then* $\mathrm{NP}_{(K,+,-,=)} \subseteq \mathrm{P}_{(K,+,\times,=)}$.

In other words, separating P and NP over K thanks to a problem of $\mathrm{NP}_{(K,+,-,=)}$ (such as TwentyQuestions) requires to separate VP and VNP.

VP and VNP (2)

Corollary

If $\mathrm{VP} = \mathrm{VNP}$ *then* $\mathrm{NP}_{(K,+,-,=)} \subseteq \mathrm{P}_{(K,+,\times,=)}$.

In other words, separating P and NP over K thanks to a problem of $\mathrm{NP}_{(K,+,-,=)}$ (such as TwentyQuestions) requires to separate VP and VNP.

. . . What about the multiplication in NP?

- Requires to handle hypersurfaces instead of hyperplanes.
- Algorithms work in PSPACE only.
- $\rightarrow$ go beyond NP_K and VNP.

Outline

PSPACE in BSS model

- $\mathrm{PAR}_{\mathbb{R}}$ (parallel polynomial time): languages $A \subseteq \cup_{n\geq 0}\mathbb{R}^n$ recognized by polynomial-depth algebraic circuits.

PSPACE in BSS model

- $\mathrm{PAR}_{\mathbb{R}}$ (parallel polynomial time): languages $A \subseteq \cup_{n \geq 0} \mathbb{R}^n$ recognized by polynomial-depth algebraic circuits.

- Example: Reachability in a semialgebraic set – Let S be a semialgebraic set given by a boolean combination of inequations and let $s, t \in S$. Are s and t in the same connected component of S?

PSPACE in Valiant's model

- VPSPACE: families of polynomials computed by polynomial-depth arithmetic circuits.

PSPACE in Valiant's model

- VPSPACE: families of polynomials computed by polynomial-depth arithmetic circuits.
- Example: The resultant of a system of polynomials. (Expressible as a quotient of determinants of easy-to-compute matrices of exponential size)

Alternative definitions of VPSPACE

- Families of polynomials whose coefficients are computable in PSPACE.

Alternative definitions of VPSPACE

- Families of polynomials whose coefficients are computable in PSPACE.
- Generalization of VNP: instead of a summation at the end of a polynomial-size arithmetic circuit, we can use summation gates in the circuits (Poizat 2006).

Does VPSPACE have small circuits?

Theorem (nonuniform setting)

$\mathrm{VP} = \mathrm{VPSPACE}$ *if and only if* $\mathrm{P} = \mathrm{PSPACE}$ *and* $\mathrm{VP} = \mathrm{VNP}$.

Does VPSPACE have small circuits?

Theorem (nonuniform setting)

$\mathrm{VP} = \mathrm{VPSPACE}$ *if and only if* $\mathrm{P} = \mathrm{PSPACE}$ *and* $\mathrm{VP} = \mathrm{VNP}$.

Theorem (uniform setting)

$\mathrm{VP} = \mathrm{VPSPACE}$ *implies* $\mathrm{PSPACE} = \mathrm{P}$–uniform NC.

Does VPSPACE have small circuits?

Theorem (nonuniform setting)

$\mathrm{VP} = \mathrm{VPSPACE}$ *if and only if* $\mathrm{P} = \mathrm{PSPACE}$ *and* $\mathrm{VP} = \mathrm{VNP}$.

Theorem (uniform setting)

$\mathrm{VP} = \mathrm{VPSPACE}$ *implies* $\mathrm{PSPACE} = \mathrm{P}$–uniform NC.

Theorem

If symmetry of information for polylogarithmic space Kolmogorov complexity holds true, then $\mathrm{VP} \neq \mathrm{VPSPACE}$.

Transfer over the complex field

Theorem

If $\mathrm{VP} = \mathrm{VPSPACE}$ *then* $\mathrm{PAR}_{\mathbb{C}} = \mathrm{P}_{\mathbb{C}}$.

Transfer over the complex field

Theorem

If $\mathrm{VP} = \mathrm{VPSPACE}$ *then* $\mathrm{PAR}_{\mathbb{C}} = \mathrm{P}_{\mathbb{C}}$.

Proof.

Same idea as for big products (i.e. point location in a set of cells)

Theorem

If $\mathrm{VP} = \mathrm{VPSPACE}$ *then* $\mathrm{PAR}_{\mathbb{C}} = \mathrm{P}_{\mathbb{C}}$.

Proof.

Same idea as for big products (i.e. point location in a set of cells), but:

- Cell in an arrangement of hypersurfaces;

Theorem

If $\mathrm{VP} = \mathrm{VPSPACE}$ *then* $\mathrm{PAR}_{\mathbb{C}} = \mathrm{P}_{\mathbb{C}}$.

Proof.

Same idea as for big products (i.e. point location in a set of cells), but:

- Cell in an arrangement of hypersurfaces;
- For the binary search, tests of membership to a variety given by an exponential number of polynomials:
  - express it by $n + 1$ polynomials only thanks to transcendental coefficients,
  - replace these coefficients by integers growing sufficiently fast.

□

Transfer over the reals (1)

Theorem

If $\mathrm{VP} = \mathrm{VPSPACE}$ *then* $\mathrm{PAR}_{\mathbb{R}} = \mathrm{P}_{\mathbb{R}}$.

Transfer over the reals (1)

Theorem

If $\mathrm{VP} = \mathrm{VPSPACE}$ *then* $\mathrm{PAR}_{\mathbb{R}} = \mathrm{P}_{\mathbb{R}}$.

Same idea as for $\mathbb{C}$

Transfer over the reals (1)

Theorem

If $\mathrm{VP} = \mathrm{VPSPACE}$ *then* $\mathrm{PAR}_{\mathbb{R}} = \mathrm{P}_{\mathbb{R}}$.

Same idea as for $\mathbb{C}$, but:

- Testing the membership to a variety is now trivial:

$$(\forall i, f_i(\bar{x}) = 0) \Leftrightarrow \sum f_i(\bar{x})^2 = 0;$$

Transfer over the reals (1)

Theorem

If $\mathrm{VP} = \mathrm{VPSPACE}$ *then* $\mathrm{PAR}_{\mathbb{R}} = \mathrm{P}_{\mathbb{R}}$.

Same idea as for $\mathbb{C}$, but:

- Testing the membership to a variety is now trivial:

$$(\forall i, f_i(\bar{x}) = 0) \Leftrightarrow \sum f_i(\bar{x})^2 = 0;$$

- Taking sign tests into account is more cumbersome: we use the following lemma (nonconstructive version due to Grigoriev, 2000): Let $u_1, \ldots, u_N$ be distinct vectors of ${\mathbb{F}_2}^d$.
  - There exists a vector $v \in {\mathbb{F}_2}^d$ such that

$$\frac{N}{2} - \frac{\sqrt{N}}{2} \leq \#\{i \mid u_i \cdot v = 0\} \leq \frac{N}{2} + \frac{\sqrt{N}}{2}.$$

  - Such a vector can be found in logarithmic space.

Transfer over the reals (2)

How to use the constructive version of Grigoriev's lemma?

Goal: binary search among the cells.

- The cell of x, viewed as a vector $S \in \{0,1\}^s$, satisfies

$$u.S = 1 \iff \prod_{i|u_i=1} f_i(\bar{x}) < 0$$

Transfer over the reals (2)

How to use the constructive version of Grigoriev's lemma?

Goal: binary search among the cells.

- The cell of x, viewed as a vector $S \in \{0,1\}^s$, satisfies

$$u.S = 1 \iff \prod_{i|u_i=1} f_i(\bar{x}) < 0$$

- If $u \in \{0,1\}^s$ is orthogonal to roughly half the cells, then roughly half of them are discarded at each step.

Transfer over the reals (2)

How to use the constructive version of Grigoriev's lemma?

Goal: binary search among the cells.

- The cell of x, viewed as a vector $S \in \{0,1\}^s$, satisfies

$$u.S = 1 \iff \prod_{i|u_i=1} f_i(\bar{x}) < 0$$

- If $u \in \{0,1\}^s$ is orthogonal to roughly half the cells, then roughly half of them are discarded at each step.

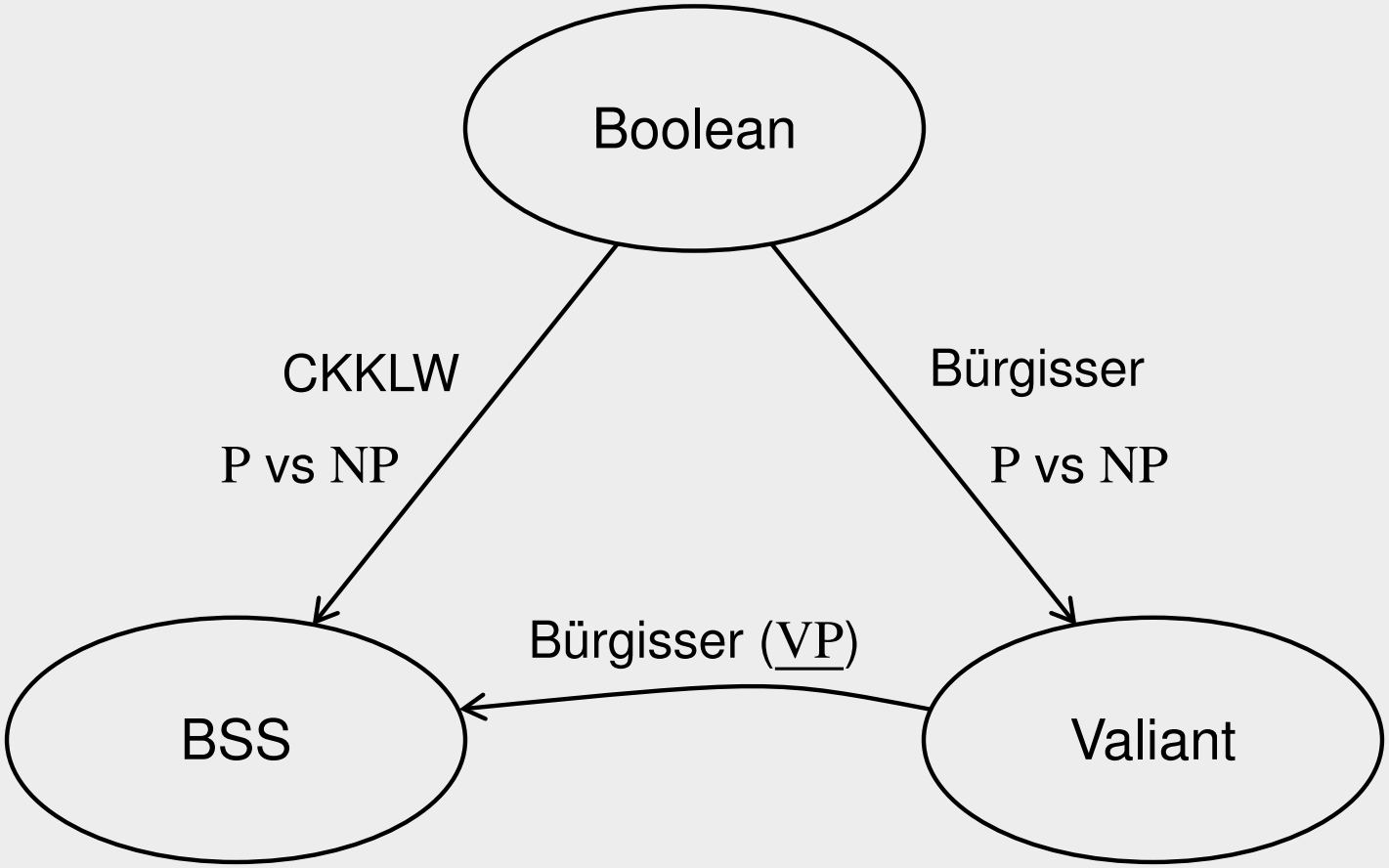
- Simply exponential number of cells $\rightarrow$ polynomial-time algorithm.

□

Outline

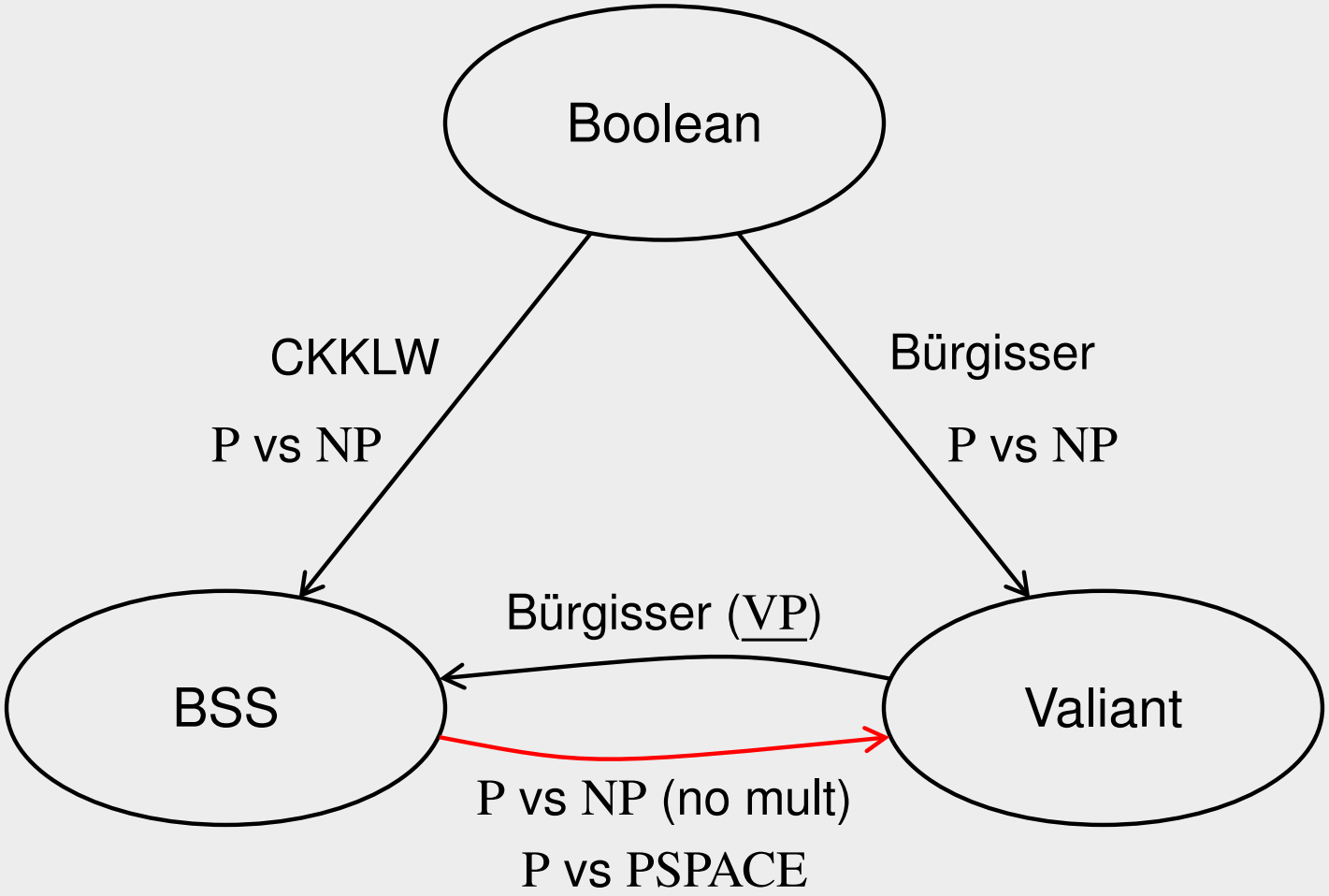
The picture

Separations of classes in different models.



The picture

Separations of classes in different models.



Perspectives

- Converse of the transfer theorems?

Perspectives

- Converse of the transfer theorems?
- More generally, complete the previous picture.

Perspectives

- Converse of the transfer theorems?
- More generally, complete the previous picture.

- Natural complete family in VPSPACE?

Perspectives

- Converse of the transfer theorems?
- More generally, complete the previous picture.

- Natural complete family in $\mathrm{VPSPACE}$?

- Separation of $\mathrm{VPSPACE}$ and VP?
  And of PSPACE and P-uniform NC?

References

Cooking pasta:

`http://www.trucmania.com/Recettes/Cuisson-des-pates.html`

Mille-feuille recipe:

`http://www.cuisinorama.com/recettes/mille_feuille_493.html`